Implementation of Graph DBMS for Road Networks and Points of Interest

Zhenmu Gong
Tsu-Hao Fu
Noah Hung

Literature Review

1. **GRFusion**
- Implement a graph data structure using VoltDB as the underlying RDBMS.
- Allow graph operator and relational operator to be co-existed in the same query execution pipeline.
- Create a view of the relational data in the graph model.
- Perform graph traversal operations efficiently over native graph representation.
- Extend the SQL language of VoltDB to declaratively find paths in graph views.

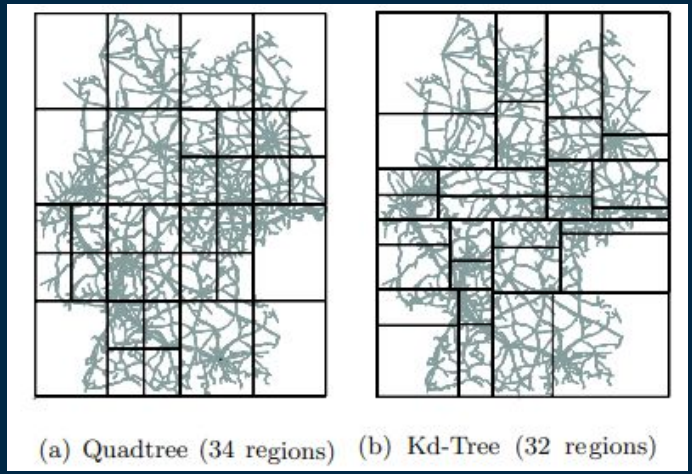
2. **Partitioning Graphs to Speed Up Dijkstra's (Tested but not used in our simulation)**
3. **Quadtree and R-tree indexes in Oracle Spatial: A Comparison using GIS Data (We used R-tree as it has faster neighbours queries and window queries.)**
4. **Engineering Label-Constrained Shortest-Path Algorithms (inspiration shortest path with label constraint)**

Literature Review

Partitioning Graphs to Speed Up Dijkstra's
(Not used in our simulation)

Comparison between multiple graph partition data structure such as Grid, KdTree or METIS with different number of partitions. All partitions are faster than plain Dijkstra's with a minimal speed-up of more than 50.

The speed-up increases as the size of graph increase.



Rolf H. Möhring Technische Universität Berlin, Möhring, R. H., Berlin, T. U., Berlin, H. S. T. U., Schilling, H., (TH), B. S. U. K., Schütz, B., (TH), U. K., (TH), D. W. U. K., Wagner, D., (TH), T. W. U. K., Willhalm, T., Contributor MetricsExpand All Rolf H Möhring Technical University of Berlin Publication Years1984 - 2019Pub, & Rolf H Möhring Technical University of Berlin Publication Years1984 - 2019Publication counts44Available for Download5Citation count638Downloads (cumulative)4. (1970, January 1). *Partitioning graphs to speedup Dijkstra's algorithm*. ACM Journal of Experimental Algorithmics. Retrieved April 1, 2023, from https://dl.acm.org/doi/pdf/10.1145/1187436.1216585

Literature Review

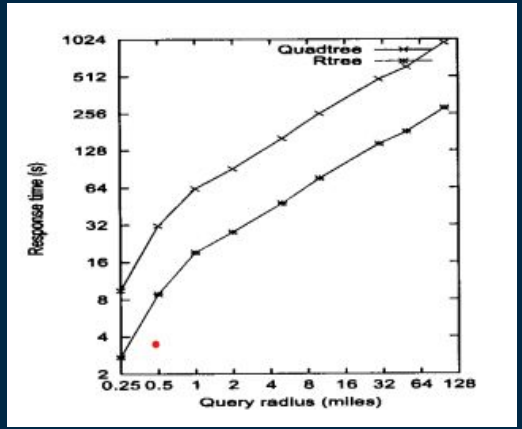
Quadtree and R-tree indexes in Oracle Spatial: A Comparison using GIS Data

They used their optimizatized implementation to test both data structures.

The final result is that R tree outperforms Quadtree

Quadtree are only created faster But it requires fine-tuning by choosing appropriate tiling level.

Thus, we chose R tree as it has faster neighbours queries and window queries.



Ravi Kanth V Kothuri Oracle Corporation, Kothuri, R. K. V., Corporation, O., Corporation, S. R. O., Ravada, S., Corporation, D. A. O., Abugov, D., Madison, U. of W.-, California, U. of, & Metrics, O. M. V. A. (2002, June 1). Quadtree and R-tree indexes in Oracle Spatial: Proceedings of the 2002 ACM SIGMOD international conference on management of data. ACM Conferences. Retrieved March 31, 2023, from https://dl.acm.org/doi/10.1145/564691.564755

Environment Setup

1. Programming Languages: Python, SQL

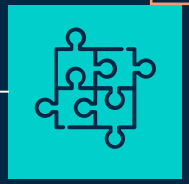
2. Databases:

- SQLite/Dataframe/Python Dictionary: Implementation and Simulation
- PostgreSQL/PostGIS: Validation and Visualization
- GRFusion: (Abandoned)

3. Data Processing Tool: Osimium, Osm2pgsql

4. Final implementation is based on python dictionary for database simulation.

TABLE OF CONTENTS



01

Dataset & Preprocessing

Import OSM file into simulated graph database

02

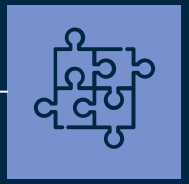
Development

Three main functions for GPS such as shortest path, range query, and nearest neighbors

03

Simulation

Design query language for these functions and validate all features based on West Lafayette map



04

Conclusion & Future work

To summarize our work and rooms to improve in the future

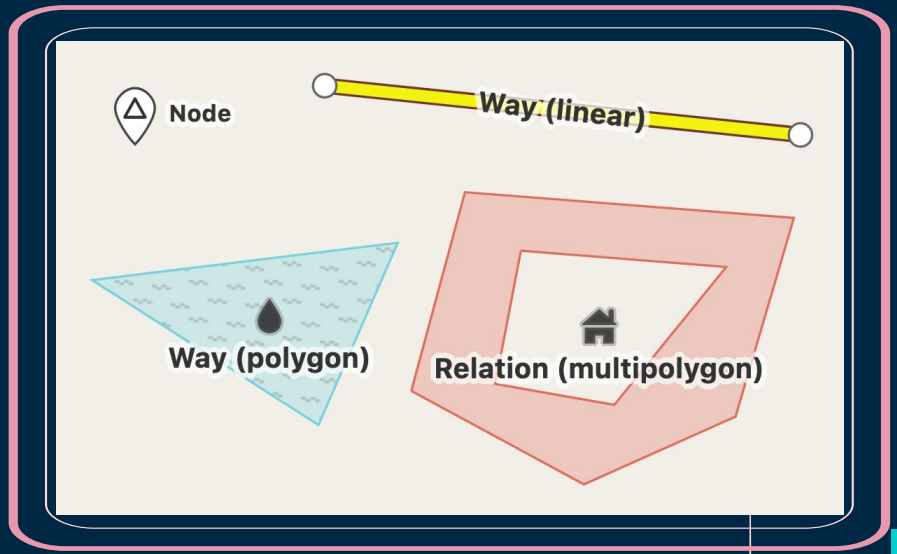
01 - Dataset and Preprocessing (PHASE 01)

OpenStreetMap

OSM uses a topological data structure, with four core elements:

- **Node**: points with latitude/longitude.
- **Way**: lines or polygons.
- **Relation**: relationships of existing nodes and ways

Our road network focuses on Node and Way



1.OSM data in PBF files ,by OpenStreepMap (2023, March,1) Retrieved from:https://extract.bbbike.org/
2.OSM elements example, by Wikipedia (2023, May, 3) Retrieved from:https://en.wikipedia.org/wiki/OpenStreetMap

01 – Dataset and Preprocessing (PHASE 01)

1. Extract map features from the source data (.pbf) by using **Osimium**.
2. Remove all the unwanted attributes , such as timestamp and user.
3. Filtering out nodes without interesting tags and non-road edges.
4. Decompose every edges into small segments and calculate Haversine distance for every segments.

OSM visualization in PostGIS ,by OpenStreepMap (2023, May, 1)

01 - Dataset and Preprocessing (PHASE 02)

1. Store the data as three tables into our GIS database (Simulated by **Python Dictionary**):

- poi (id, name, amenity, shop)

  ##amenity: the tags of the POI such as restaurant, grocery store, etc.
  ##shop: also the tags of POI

- nodes (id, lon, lat)

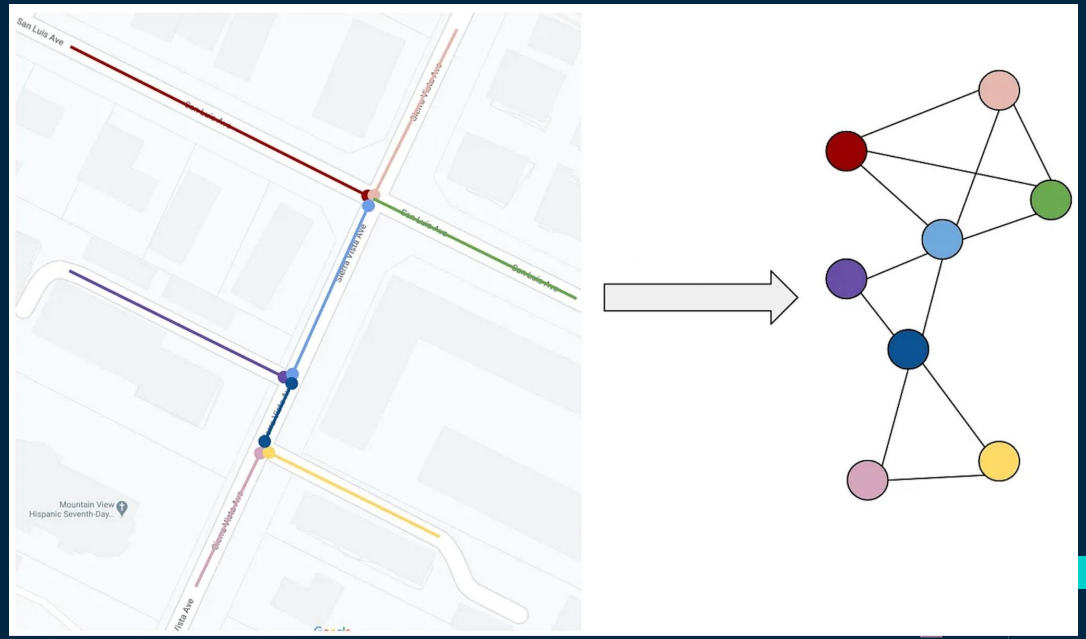
  ##lon: longitude
  ##lat: latitude

- edges (id, wid, start, end, distance, highway)

  ##wid: id for the edges
  ##highway: whether the edge is driving-allowed

01 - Dataset and Preprocessing (PHASE 03)

1. Build the road network graph based on the nodes and edges tables.
2. Create R-tree indexes for the road network and the POIs.



1.Road network graph ,by Adit Jain (2022, Jan,5) Retrieved from: https://medium.com/@aditiajain/analyzing-openstreetmap-road-network-attributes-with-networkx-pyg-and-graph-neural-networks-2f3d7b0f832

01 - Dataset and Preprocessing (PHASE 04)

1. PostGIS is an extension for the PostgreSQL database. It adds additional geospatial types and functions which make handling spatial data in the database easier and more powerful.

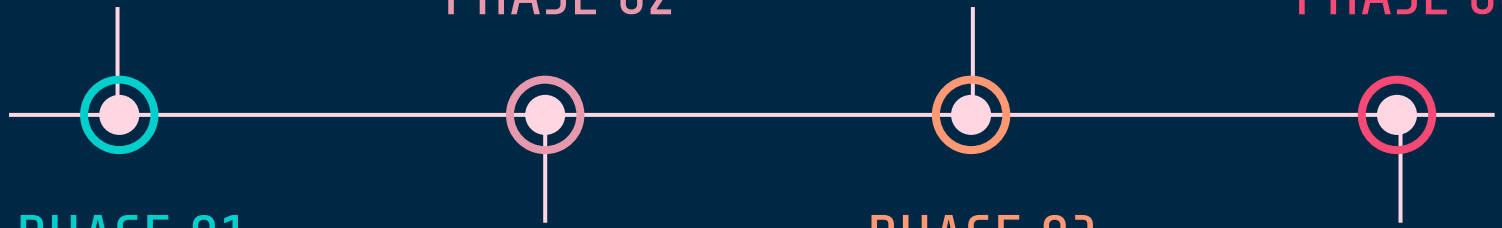
2. We use PostGIS as the baseline to evaluate our data-preprocessing and query performance.

1.PostGIS Logo ,by Wikipedia (2023, May, 1)
Retrieved from: https://en.wikipedia.org/wiki/PostGIS

02 - Development

1. Procedure

Store Geo-data into R-tree structure for future use and construct graph



PHASE 01

Data Structure

PHASE 02

Shortest Path

Given road network, find the shortest path between two nodes

PHASE 03

K-nearest Neighbors

Given road network, find k nearest POI from starting point

PHASE 04

Range Query

Combining shortest path and KNN for searching POI within assigned miles

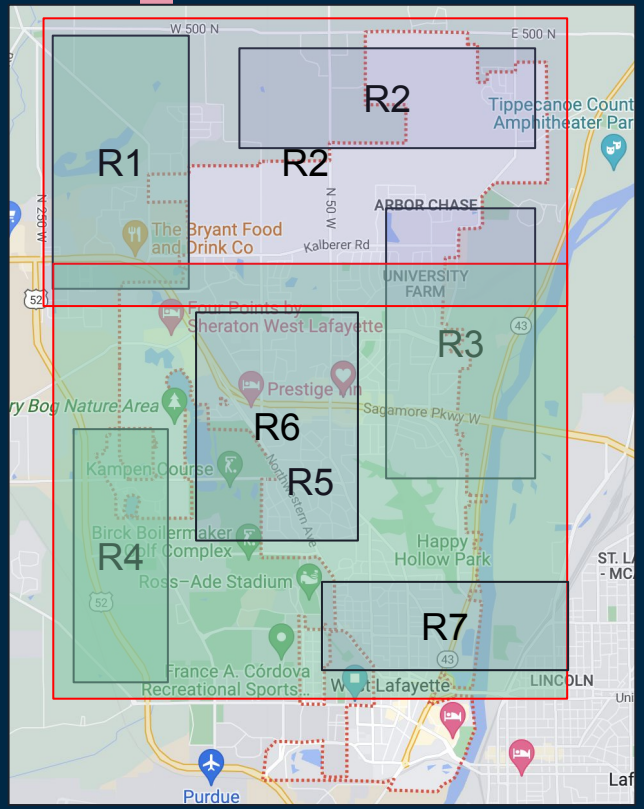
02 - Development (PHASE 01)

Data Structure

After our data preprocessing, we obtain a clean data including node table, edges table, and points of interest (POI) table.

Then, we decide to utilize R-tree as a data structure to store the spatial data index. R-tree not only stores data an efficient manner, but also highly useful for range query.

Besides, the leaf nodes of R-tree contains data related to Minimum Bounding Region (MBR), which is the bounding box surrounding the objects. In our case, the MBR is surrounding the individual edges.



02 - Development (PHASE 01)

Database

Our database is pretty functional. It's equipped with basic database function such as

1. Create table
2. Insert data
3. Update data
4. Delete data

And three gis query types:

5. Create shortestPath
6. Find K <label>
7. Find <label> in range [M,N] (in miles)

Ref:https://www.toptal.com/database/database-design-bad-practices

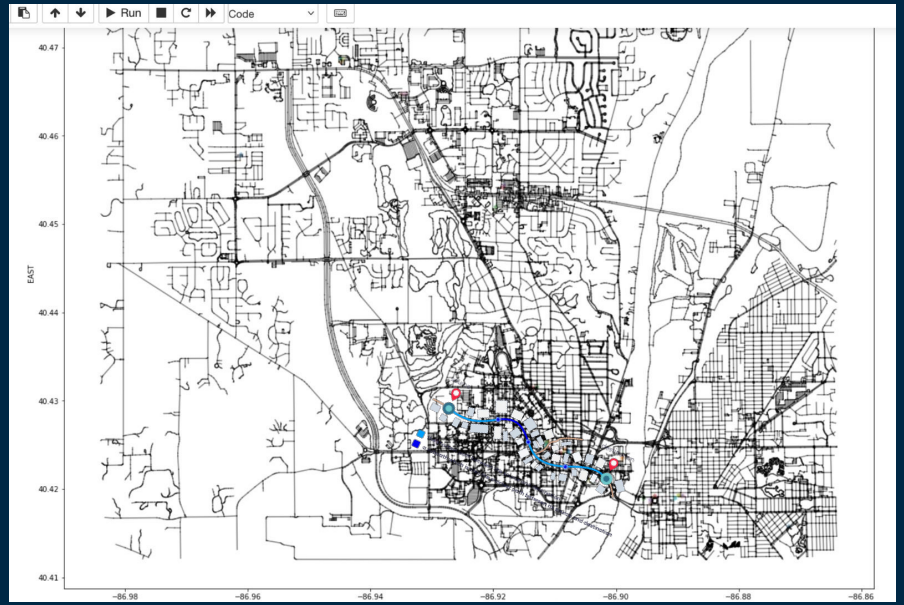
02 - Development (PHASE 02)

Shortest Path

First, we allow user to input the starting point by either id or coordinate. If it's id, we map the id with our node in database for retrieving the coordinate. If it's coordinate, we will use R-tree to map the point with the closest point among our road network. It's similar to the end point.

Then, we have implemented two methods to find the shortest path:

1. Query Language with Dikjastra's Algorithm [2]
2. Individual Function of finding shortest path with tags in DFS [5]



Ref: https://www.programiz.com/dsa/dijkstra-algorithm

02 - Development (PHASE 03)

K-nearest neighbors

We decide to utilize R-tree as a data structure to store the spatial data index. Then we apply the embedded nearest neighbor function to find the closest k neighbors from the starting point.

As what we mentioned above, we get the mapping point on roadmap by R-tree.

The right-hand side figure display the closest restaurants from Wabashing Landing. User can choose the searching tags based on their needs.

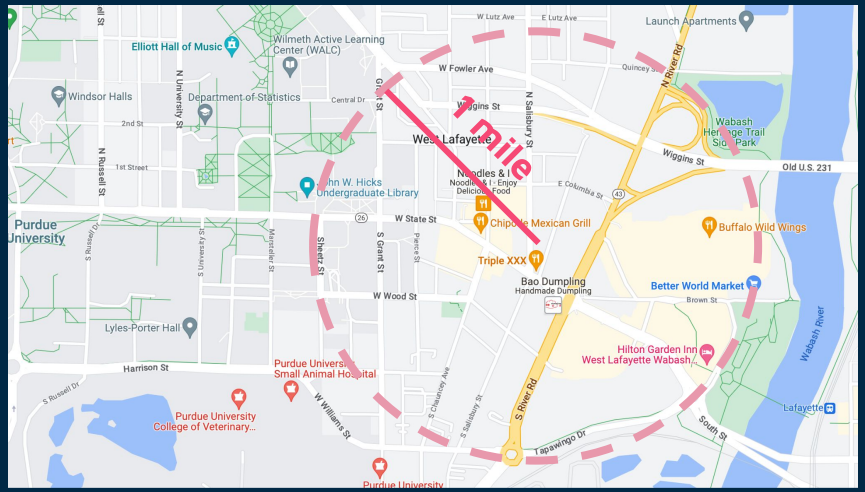
Labels on the map is restaurants near Wabash Landing

02 - Development (PHASE 04)

Range Query

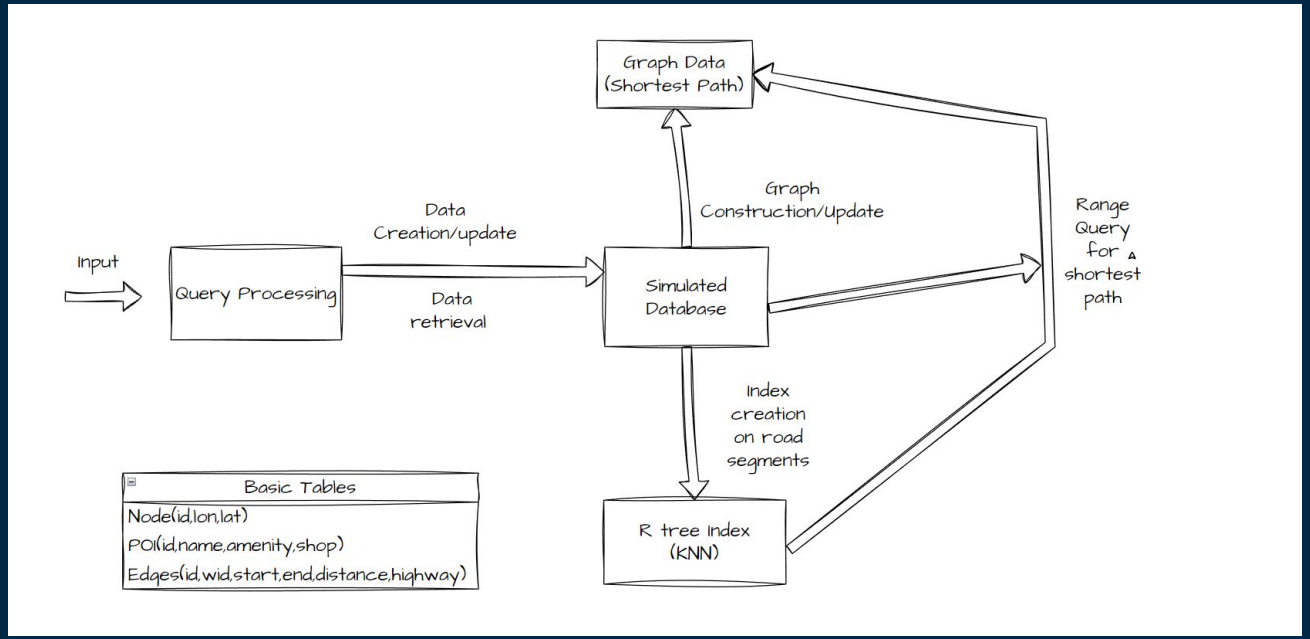
Our range query is a combination of the Shortest-path and KNN functions. It retrieves all the poi of a given tag and find the poi which their shortest-path length from start point are within a specified range.

As shown in the plot, if we want to find coffee shops around Wabash Landing within 1 mile. We will retrieve all the coffee shops in our POI table and filtered those out of our desired range.



03 - Simulation

Diagram



03 - Simulation

Target:

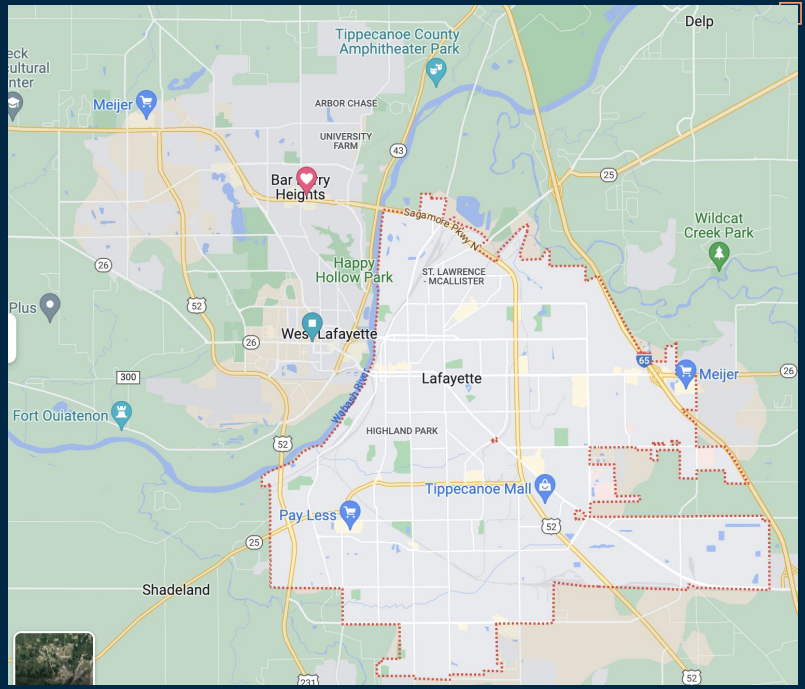
In the simulation, we will let user input required commands to implement some functions.

Data Loading:
'load file': Input the pathname of pbf file

Three Main Functions:
`Create shortest path from id1 to id2'
`Create shortest path from [lat1, lon1] to [lat2, lon2]'

`Find k nearest restaurants from [lat1, lon1]'

`Find restaurants within x miles'



03 - Simulation

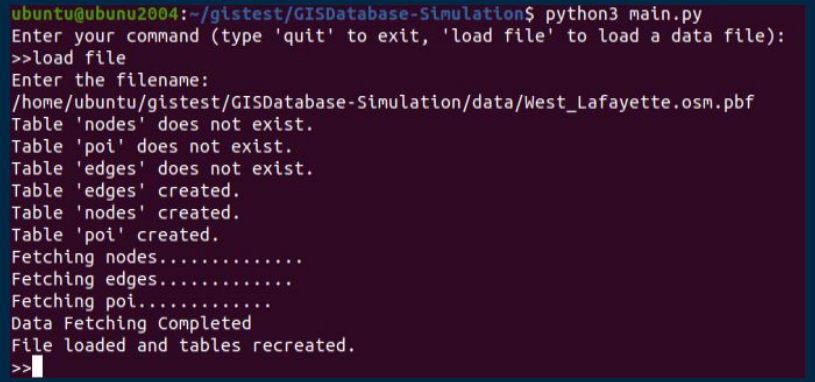
Data Loading:

If Data Loading is successfully executed, it'll show

CREATE TABLE poi(id, name, amenity, shop)

CREATE TABLE nodes(id, lon, lat)

CREATE TABLE edges(id, wid, start, end, distance, highway)



03 – Simulation

View Data:

View POI Table

```
>>select * from poi
```

id	name	amenity	shop
3402051224		bench	
3056730279	earth	atmospheric	and planetary sciences library
3538939202		parking_entrance	
6260876375		bicycle_parking	
3588527857		bench	
9964376995		bench	
9525991454	discount den		convenience
3056718619	siegesmund engineering library	library	
8256139023	chai tae kwon do	dojo	
3656165471		waste_basket	
3056718617	humanities	social sciences and education (hsse) library	library
9995853713		bench	
8772656845	culver's	fast_food	
9998555938		waste_basket	
8000961897	jimmy john's	fast_food	
3107803395	java roaster	cafe	
10163919930		bench	
10163919933		bench	
8752737642	wood (elizabeth g. and william r.) residence hall		
9518326635	meijer		convenience
586882736	cook research	inc.	
3394220030		bench	
10163849108		parking	
8004006746	little free library	public_bookcase	
9178858245	dawn redwood (bg)		
9974087865		parking_entrance	
3684847368		bench	
9609847823	welcome to lafayette		
10036636304		waste_basket	
9178789466	rhodo		
9964959817		waste_basket	
9178797758	hazelnut		
10145452880	dr. guthrie carr orthodontist	dentist	
10163849040	kung fu tea	cafe	
6260876376		bicycle_parking	
153560736	lafayette		
10145452878		fast_food	
9776704352	dos amigos mexican restaurant	restaurant	
3394220037		bench	
846238607	happy hollow bus602w		
6955583701	carlisle rd & windsor dr (nw corner) bus623nw		
358664225	purdue university dairy farm	school	

Filtering data

Total rows: 242

```
>>select * from nodes where id > 10595366804 and lon < -86.94
```

id	lon	lat
10650872370	-86.9656896	40.4700944
10650872371	-86.9655601	40.4700961
10650872372	-86.965565	40.4703111
10650872373	-86.9656945	40.4703094
10650872374	-86.965861	40.4700919
10650872375	-86.9657315	40.4700936
10650872376	-86.9657363	40.4703086
10650872377	-86.9658659	40.4703069
10650872378	-86.9660357	40.4700901
10650872379	-86.9659061	40.4700918
10650872380	-86.965911	40.4703068
10650872381	-86.9660405	40.4703051

Total rows: 12

The id of POI is a subset of node id in nodes table.(Join operation can be used to get coordinates for poi)

03 - Simulation

Query examples:

create shortestpath from 358665942 to 9519034907

create shortestpath from [-86.8801548,40.4246338] to [-86.8868154,40.4200306]

create shortestpath from 10087689850 to 9178785027 where service < 0.1

find restaurant from 358665942 where distance between 0 and 5

find restaurant from [-86.9359556,40.4365242] where distance between 2 and 4

find 5 restaurant from [-86.8945389,40.4183154]

find 7 library from 9178814647

find 2 bicycle_parking from 9844603592

03 - Simulation

Shortest Path:

Use command >> create shortestpath from 358665942 to 9519034907
Return table with showing the path with several informative attributes.

```
File loaded and tables recreated.
create shortestpath from 358665942 to 9519034907
Total distance: 3.9870421150844733 meters
```

start_node_id	start_node_lon	start_node_lat	end_node_id	end_node_lon	end_node_lat	distance	highway_type
9205092445	-86.8653957	40.4743809	1302793233	-86.8658199	40.4744839	0.023403805914645266	
1302793233	-86.8658199	40.4744839	1302793202	-86.8664583	40.4746864	0.03635389892348293	
1302793202	-86.8664583	40.4746864	1302793100	-86.8669537	40.4748477	0.028322502296837755	
1302793100	-86.8669537	40.4748477	1302793260	-86.8678139	40.475082	0.048021874578484613	
1302793260	-86.8678139	40.475082	8041554207	-86.8679776	40.4748202	0.020030621491868697	
8041554207	-86.8679776	40.4748202	8041554208	-86.8697129	40.4742585	0.09911942566583286	
8041554208	-86.8697129	40.4742585	8041554213	-86.8718186	40.4695616	0.3428789427103075	
8041554213	-86.8718186	40.4695616	8041554212	-86.8724704	40.4674476	0.15002786475982308	
8041554212	-86.8724704	40.4674476	8041554215	-86.872992	40.4654318	0.14195159437068197	
8041554215	-86.872992	40.4654318	8041554216	-86.8738737	40.4619427	0.24548891290648833	
8041554216	-86.8738737	40.4619427	8038742596	-86.8740149	40.4619681	0.00762736442602717	service
8038742596	-86.8740149	40.4619681	38021812	-86.874137	40.461507	0.03249913400123939	tertiary
38021812	-86.874137	40.461507	8038742601	-86.8741744	40.4613484	0.011133194166549273	tertiary
8038742601	-86.8741744	40.4613484	8041311482	-86.8741805	40.4613227	0.0018044236306535464	tertiary
8041311482	-86.8741805	40.4613227	38021818	-86.8743501	40.4606028	0.05053310043904772	tertiary

03 - Simulation

Shortest Path:

Use command
>> create shortestpath from 10087689850 to 9178785027 where service < 0.1
Return table of the path with a limited usage of certain road (in miles).

```
>>create shortestpath from 10087689850 to 9178785027  where service < 0.1
Total distance: 3.08 miles
```

start_node_id	start_node_lon	start_node_lat	end_node_id	end_node_lon	end_node_lat	distance	highway_type
8680003943	-86.9210825	40.4269917	8680003944	-86.9210717	40.4269834	0.001	
8680003944	-86.9210717	40.4269834	8680003945	-86.9210831	40.4269749	0.001	
8680003945	-86.9210831	40.4269749	8679988370	-86.9211001	40.4269623	0.001	
8679988370	-86.9211001	40.4269623	8679988369	-86.9210901	40.4269545	0.001	
8679988369	-86.9210901	40.4269545	1419863370	-86.9211213	40.4269313	0.002	
1419863370	-86.9211213	40.4269313	1419863244	-86.9212248	40.4269285	0.005	
1419863244	-86.9212248	40.4269285	1419863385	-86.921283	40.4268829	0.004	
1419863385	-86.921283	40.4268829	8679988362	-86.921283	40.4268523	0.002	
8679988362	-86.921283	40.4268523	1419863394	-86.9213491	40.426729	0.009	
1419863394	-86.9213491	40.426729	8679988364	-86.9213325	40.4267163	0.001	
8679988364	-86.9213325	40.4267163	8679988363	-86.9213709	40.4266873	0.003	
8679988363	-86.9213709	40.4266873	1419863375	-86.9213246	40.4266519	0.003	
1419863375	-86.9213246	40.4266519	8679988405	-86.9213511	40.4266317	0.002	
8679988405	-86.9213511	40.4266317	8679988409	-86.9213578	40.4266368	0.0	
8679988409	-86.9213578	40.4266368	8679988408	-86.9213893	40.4266129	0.002	
8679988408	-86.9213893	40.4266129	8679988407	-86.9213947	40.4266171	0.0	

03 - Simulation

K-nearest neighbors:

```
>>find 5 restaurant from [-86.8945389,40.4183154]
+------------+---------------+------------+------+------------+-----------+----------+
| id         | name          | amenity    | shop | lon        | lat       | distance |
+------------+---------------+------------+------+------------+-----------+----------+
| 9784444505 | merlin's beard | restaurant |      | -86.8945389 | 40.4183154 | 0.0      |
| 2873883552 | chumlies      | restaurant |      | -86.8944521 | 40.4188013 | 0.034    |
| 2873883553 | la scala      | restaurant |      | -86.8938529 | 40.4192649 | 0.075    |
| 10163919946 | bru burger bar | restaurant |      | -86.8956236 | 40.4189775 | 0.073    |
| 5290416854 | pete's diner  | restaurant |      | -86.8904566 | 40.4189729 | 0.219    |
+------------+---------------+------------+------+------------+-----------+----------+
Total rows: 5
```

03 - Simulation

Range Query:

```
>>find restaurant from 358665942 where distance between 0 and 1.5
+-------------+----------------------------------------+------------+------+-------------+------------+----------+
| id          | name                                   | amenity    | shop | lon         | lat        | distance |
+-------------+----------------------------------------+------------+------+-------------+------------+----------+
| 2873883573  | lafayette brewing co.                  | restaurant |      | -86.8899454 | 40.4193119 | 1.282    |
| 5286685864  | dt kirby's                             | restaurant |      | -86.8893824 | 40.4192745 | 1.266    |
| 5296582823  | professor joe's sports bar & pizzeria  | restaurant |      | -86.889145  | 40.4192483 | 1.26     |
| 4412612599  | parkside seafood house                 | restaurant |      | -86.8719733 | 40.4158218 | 1.267    |
| 5290416854  | pete's diner                           | restaurant |      | -86.8904566 | 40.4189729 | 1.317    |
| 9784444505  | merlin's beard                         | restaurant |      | -86.8945389 | 40.4183154 | 1.492    |
| 2873883552  | chumlies                               | restaurant |      | -86.8944521 | 40.4188013 | 1.464    |
| 2873883553  | la scala                               | restaurant |      | -86.8938529 | 40.4192649 | 1.419    |
| 10163919946 | bru burger bar                         | restaurant |      | -86.8956236 | 40.4189775 | 1.499    |
| 9776704352  | dos amigos mexican restaurant          | restaurant |      | -86.8991776 | 40.4235634 | 1.459    |
+-------------+----------------------------------------+------------+------+-------------+------------+----------+
Total rows: 10
```

04 - Conclusion and Future Work

Conclusion:

1. As a topic of graph database, we do learn how to implement spatial data on graph.
2. According to our code, it's pretty flexible for users to customize their POI table, nodes table, and edges table.
3. Due to the limitation of computational resource, our GPS scope is based on West Lafayette and Lafayette. The code should be capable of handling any osm files.
   (Everything is executed on Colab)
4. Shortest Path/ Range Query/ KNN are all functional.
5. Our functions are **versatile**.
   Ex: we can limit the shortest path to avoid roads with certain tags.

04 – Conclusion and Future Work

Challenges:

1. In the first milestone, we are stuck in the implementation of GRFusion, and we spent plenty of time to read documents.
2. The complexity of Topological data structure while handling the spatial data from OSM.

Future Work:

1. If we obtain more computational resource, we will try to expand the scale, such as Chicago or New York.
2. We hope to combine other indexing methods and our methodology to re-implement this project for comparison.
3. Add visualization function for better user experience.

Reference(1)

University, Mohamed S. Hassan Purdue, et al. “Grfusion: Proceedings of the 2018 International Conference on Management of Data.” *ACM Conferences*, 1 May 2018, https://dl.acm.org/doi/abs/10.1145/3183713.3193541.

Gao, Jun, et al. “Relational Approach for Shortest Path Discovery over Large Graphs.” *ArXiv.org*, 31 Dec. 2011, https://arxiv.org/abs/1201.0232.

OpenStreetMap, https://www.openstreetmap.org/#map=13/40.4272/-86.9479&layers=N.

Ravi Kanth V Kothuri Oracle Corporation, Kothuri, R. K. V., Corporation, O., Corporation, S. R. O., Ravada, S., Corporation, D. A. O., Abugov, D., Madison, U. of W.-, California, U. of, & Metrics, O. M. V. A. (2002, June 1). *Quadtree and R-tree indexes in Oracle Spatial: Proceedings of the 2002 ACM SIGMOD international conference on management of data*. ACM Conferences. Retrieved March 31, 2023, from https://dl.acm.org/doi/10.1145/564691.564755

Reference(2)

Rolf H. Möhring Technische Universität Berlin, Möhring, R. H., Berlin, T. U., Berlin, H. S. T. U., Schilling, H., (TH), B. S. U. K., Schütz, B., (TH), U. K., (TH), D. W. U. K., Wagner, D., (TH), T. W. U. K., Willhalm, T., Contributor Metrics Expand All Rolf H Möhring Technical University of Berlin Publication Years1984 - 2019Pub, & Rolf H Möhring Technical University of Berlin Publication Years1984 - 2019Publication counts44Available for Download5Citation count638Downloads (cumulative)4. (1970, January 1). *Partitioning graphs to speedup Dijkstra's algorithm*. ACM Journal of Experimental Algorithmics. Retrieved April 1, 2023, from https://dl.acm.org/doi/pdf/10.1145/1187436.1216585

Do you have any questions?

hung68@purdue.edu
gong133@purdue.edu
fu390@purdue.edu

THANKS